Visual Programming

1. **What is Visual Basic?**
   Visual Basic is a powerful application development toolkit developed by John Kemeny and Thomas Kurtz. It is a Microsoft Windows Programming language.
   Visual means methods used to create the GUI using in-built objects.
   Basic refers to the basic language used in the form of basic syntax statements.

2. **What are different Editions available in VB?**
   Visual basic is available in 3 editions: Standard, Professional and Enterprise

Standard – Creates simple windows application. Basic controls are setup, icons and help files.
Professional – Used by Computer professionals, it includes Active X and internet controls.
Enterprise – Most advanced edition used to build applications in a team environment, it includes features like OLE automation and Remote Database access tools.

3. **List the various features of VB**
   The features of VB are as follows:
   1. **Client- Server Architecture** – It supports Client- Server Architecture, ActiveX, COM (Component Object Model), DCOM (Distributed Component Object Model) and ODBC (Open Data-Base Connectivity).
   2. **IntelliSense** – This feature enables the system to interact with user during coding. Few IntelliSense features are:
      a) Quick Info – Displays syntax in a Tool Tip like window.
      b) Complete Word –Automatically completes the word after enough characters have been typed.
      c) Data Tips – A yellow Tool Tip that displays values at run time.
      d) List Members - The properties of a given object are shown.
      e) List Constants – It lists all the constants of an object.
   3. **IDE (Integrated Development Environment)** – Multiple projects can be done at the same time, hence saving time for coding and debugging.
   4. **ADODC (ActiveX DataObject DataControl)**- It allows to create database application with codes.
   5. **Data List, Data Combo Controls**–These controls contain data which can be linked to ADODC in the form of a List Box or a Combo box.
   6. **Data Time Picker Control** – It allows to pick date and time.
   7. **Data Report** – It is used to create reports from any record set.
   8. **IIS (Internet Information Server) Applications** – The user can write Server side Internet application which use VB 6.0
   9. **Hierarchical Flex Grid, Data Grid Control –** Records can be displayed as a separate row within the grid using both Grids.
   10. **Bookmarks** – This feature is placed for quick search.
   11. **Property Window** – Made to list properties alphabetically or by category.
   12. **Mouse usage**: i. This feature toggles breakpoints.
       ii. All highlighted code can be commented or uncommented.

4. **What is IDE? What are the types of applications that can be created in VB**
   Visual Basic is an Integrated Development Environment (IDE). The most important applications that can be developed in Visual Basic are:

- **Standard EXE:**Standard EXE application is common to all versions of Visual Basic.
- **ActiveX EXE, ActiveXDLL:**Both are code building components which do not have a visible interface, they exhibit identical functions but both are packaged into different Dynamic Link Libraries or executable files.
- **ActiveX Control:**This application is used to build own custom controls also called OCX file.
- **VB Application Wizard:**VB Application Wizard project is used to provide a new skeleton for an application (similar to word processor template).
- **VB Wizard Manager:**VB Wizard Manager is used to build a new wizard. A wizard is a sequence of windows, which calls information from the user.
- **Data Project:**Data Project is like Standard EXE project but it adds those controls which are used in accessing database.
- **DHTML application:**DHTML Application provides facility for the development of dynamic HTML pages with the use of Visual Basic tools. It can develop those pages, which are displayed in the browser's window on a client computer.
- **Add-in:**Add-in is used for the creation of special type of commands in Visual Basic IDE. E.g. Visual Data Manager.

5. What is a Form object? Mention its events and properties

The **Form** is where the user interface is drawn. It is central to the development of Visual Basic applications.

Form Properties:

Appearance	Selects 3-D or flat appearance.
BackColor	Sets the form background color.
BorderStyle	Sets the form border to be fixed or sizeable.
Caption	Sets the form window title.
Enabled	If True, allows the form to respond to mouse and Keyboard events; if False, disables form.
Font	Sets font type, style, size.
ForeColor	Sets color of text or graphics.
Picture	Places a bitmap picture in the form.
Visible	If False, hides the form.

 Form Events:

Activate	Form_Activate event is triggered when form becomes the active window.
Click	Form_Click event is triggered when user clicks on form.
DblClick	Form_DblClick event is triggered when user double-clicks on form.

Load	Form_Load event occurs when form is loaded. This is a good place to initialize variables and set any run-time properties.

· Form Methods:

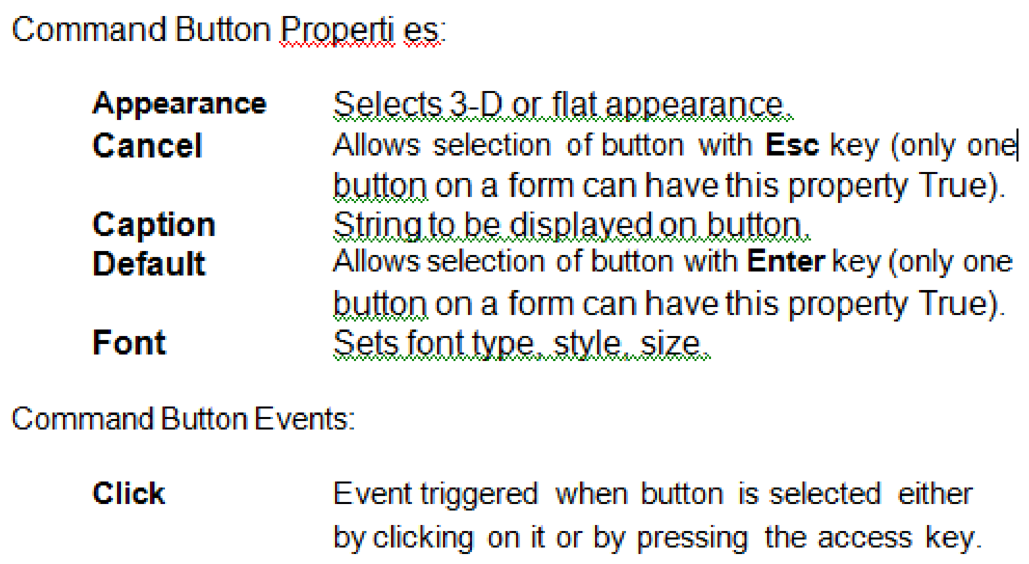
Cls	Clears all graphics and text from form. Does not clear any objects.
Print	Prints text string on the form.

Examples

frmExample.Cls ' clears the form
frmExample.Print "This will print on the form"

6. List the various intrinsic controls, its properties and methods

The **command button** is probably the most widely used control. It is used to begin, interrupt, or end a particular process.



Command Button Properti es:

Appearance	Selects 3-D or flat appearance.
Cancel	Allows selection of button with **Esc** key (only one button on a form can have this property True).
Caption	String to be displayed on button.
Default	Allows selection of button with **Enter** key (only one button on a form can have this property True).
Font	Sets font type, style, size.

Command Button Events:

Click	Event triggered when button is selected either by clicking on it or by pressing the access key.

Label Boxes

A **label box** is a control you use to display text that a user can't edit directly. We've seen, though, in previous examples, that the text of a label box can be changed at run-time in response to events.

Label Properties:

Alignment	Aligns caption within border.
Appearance	Selects 3-D or flat appearance.
AutoSize	If True, the label is resized to fit the text specifed by the caption property. If False, the label will remain the size defined at design time and the text may be clipped.
BorderStyle	Determines type of border.
Caption	String to be displayed in box.
Font	Sets font type, style, size.

Label Events:

Click	Event triggered when user clicks on a label.
DblClick	Event triggered when user double-clicks on a label.

Text Boxes

A **text box** is used to display information entered at design time, by a user at run-time, or assigned within code. The displayed text may be edited.

Text Box Properties:

Appearance	Selects 3-D or flat appearance.
BorderStyle	Determines type of border.
Font	Sets font type, style, size.
MaxLength	Limits the length of displayed text (0 value indicates unlimited length).
MultiLine	Specifies whether text box displays single line or multiple lines.
PasswordChar	Hides text with a single character.
ScrollBars	Specifies type of displayed scroll bar(s).
SelLength	Length of selected text (run-time only).
SelStart	Starting position of selected text (run-time only).
SelText	Selected text (run-time only).
Tag	Stores a string expression.
Text	Displayed text.

Text Box Events:

Change	Triggered every time the **Text** property changes.
LostFocus	Triggered when the user leaves the text box. This is a good place to examine the contents of a text box after editing.
KeyPress	Triggered whenever a key is pressed. Used for key trapping, as seen in last class.

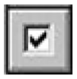
Text Box Methods:

SetFocus	Places the cursor in a specified text box.

Example

```
txtExample.SetFocus ' moves cursor to box named txtExample
```

Check Boxes



Check boxes provide a way to make choices from a list of potential candidates. Some, all, or none of the choices in a group may be selected.

Check Box Properties:

Caption	Identifying text next to box.
Font	Sets font type, style, size.
Value	Indicates if unchecked (0, vbUnchecked), checked (1, vbChecked), or grayed out (2, vbGrayed).

Check Box Events:

Click	Triggered when a box is clicked. Value property is automatically changed by Visual Basic.

Option buttons provide the capability to make a mutually exclusive choice among a group of potential candidate choices. Hence, option buttons work as a group, only one of which can have a True (or selected) value.

Option Button Properties:

Caption	Identifying text next to button.
Font	Sets font type, style, size.
Value	Indicates if selected (True) or not (False). Only one option button in a group can be True. One button in each group of option buttons should always be initialized to True at design time.

Option Button Events:

Click	Triggered when a button is clicked. **Value** property is automatically changed by Visual Basic.

List Boxes

A **list box** displays a list of items from which the user can select one or more items. If the number of items exceeds the number that can be displayed, a scroll bar is automatically added.

List Box Properties:

Appearance	Selects 3-D or flat appearance.
List	Array of items in list box.
ListCount	Number of items in list.
ListIndex	The number of the most recently selected item in list. If no item is selected, ListIndex = -1.
MultiSelect	Controls how items may be selected (0-no multiple selection allowed, 1-multiple selection allowed, 2-group selection allowed).
Selected	Array with elements set equal to True or False, depending on whether corresponding list item is selected.
Sorted	True means items are sorted in 'Ascii' order, else items appear in order added.
Text	Text of most recently selected item.

- List Box Events:

Click	Event triggered when item in list is clicked.
DblClick	Event triggered when item in list is double-clicked. Primary way used to process selection.

List Box Methods:

AddItem	Allows you to insert item in list.
RemoveItem	Removes item from list box, as identified by index of item to remove.

Examples

```
lstExample.AddItem  "This is an added item" ' adds text string to list
lstExample.Clear ' clears the list box
lstExample.RemoveItem 4 ' removes lstExample.List(4) from list box
```

Combo Boxes

The **combo box** is similar to the list box. The differences are a combo box includes a text box on top of a list box and only allows selection of one item. In some cases, the user can type in an alternate response.

Combo Box Properties:

Combo box properties are nearly identical to those of the list box, with the deletion of the MultiSelect property and the addition of a Style property.

Appearance	Selects 3-D or flat appearance.
List	Array of items in list box portion.
ListCount	Number of items in list.
ListIndex	The number of the most recently selected item in list. If no item is selected, ListIndex = -1.
Sorted	True means items are sorted in 'Ascii' order, else items appear in order added.
Style	Selects the combo box form. Style = 0, Dropdown combo; user can change selection. Style = 1, Simple combo; user can change selection. Style = 2, Dropdown combo; user cannot change selection.
Text	Text of most recently selected item.

Combo Box Events:

Click	Event triggered when item in list is clicked.
DblClick	Event triggered when item in list is double-clicked. Primary way used to process selection.

Combo Box Methods:

AddItem	Allows you to insert item in list.
Clear	Removes all items from list box.
RemoveItem	Removes item from list box, as identified by index of item to remove.

Line Tool

The **line tool** creates simple straight line segments of various width and color. Together with the shape tool discussed next, you can use this tool to 'dress up' your application.

Line Tool Properties:

BorderColor	Determines the line color.
BorderStyle	Determines the line 'shape'. Lines can be transparent, solid, dashed, dotted, and combinations.
BorderWidth	Determines line width.

There are no events or methods associated with the line tool.

Since the line tool lies in the middle-layer of the form display, any lines drawn will be obscured by all controls except the shape tool or image box.

Horizontal and Vertical Scroll Bars

Horizontal and vertical **scroll bars** are widely used in Windows applications. Scroll bars provide an intuitive way to move through a list of information and make great input devices.

Scroll Bar Properties:

LargeChange	Increment added to or subtracted from the scroll bar **Value** property when the bar area is clicked.
Max	The value of the horizontal scroll bar at the far right and the value of the vertical scroll bar at the bottom. Can range from -32,768 to 32,767.
Min	The other extreme value - the horizontal scroll bar at the left and the vertical scroll bar at the top. Can range from -32,768 to 32,767.
SmallChange	The increment added to or subtracted from the scroll bar **Value** property when either of the scroll arrows is clicked.
Value	The current position of the scroll box (thumb) within the scroll bar. If you set this in code, Visual Basic moves the scroll box to the proper position.

Shape Tool

The **shape tool** can create circles, ovals, squares, rectangles, and rounded squares and rectangles. Colors can be used and various fill patterns are available.

Shape Tool Properties:

BackColor	Determines the background color of the shape (only used when FillStyle not Solid.
BackStyle	Determines whether the background is transparent or opaque.
BorderColor	Determines the color of the shape's outline.
BorderStyle	Determines the style of the shape's outline. The border can be transparent, solid, dashed, dotted, and combinations.
BorderWidth	Determines the width of the shape border line.
FillColor	Defines the interior color of the shape.
FillStyle	Determines the interior pattern of a shape. Some choices are: solid, transparent, cross, etc.
Shape	Determines whether the shape is a square, rectangle, circle, or some other choice.

Like the line tool, events and methods are not used with the shape tool.

Picture Boxes

The **picture box** allows you to place graphics information on a form. It is best suited for dynamic environments - for example, when doing animation.

Picture boxes lie in the top layer of the form display. They behave very much like small forms within a form, possessing most of the same properties as a form.

Picture Box Properties:

AutoSize	If True, box adjusts its size to fit the displayed graphic.
Font	Sets the font size, style, and size of any printing done in the picture box.
Picture	Establishes the graphics file to display in the picture box.

Picture Box Events:

Click	Triggered when a picture box is clicked.
DblClick	Triggered when a picture box is double-clicked.

Picture Box Methods:

Cls	Clears the picture box.
Print	Prints information to the picture box.

Examples

```
picExample.Cls ' clears the box picExample
picExample.Print "a picture box" ' prints text string to picture box
```

Five types of graphics files can be loaded into a picture box:

Bitmap	An image represented by pixels and stored as a collection of bits in which each bit corresponds to one pixel. Usually has a **.bmp** extension. Appears in original size.
Icon	A special type of bitmap file of maximum 32 x 32 size. Has a **.ico** extension. We'll create icon files in Class 5. Appears in original size.

Metafile	A file that stores an image as a collection of graphical objects (lines, circles, polygons) rather than pixels. Metafiles preserve an image more accurately than bitmaps when resized. Has a **.wmf** extension. Resizes itself to fit the picture box area.
JPEG	JPEG (Joint Photographic Experts Group) is a compressed bitmap format which supports 8 and 24 bit color. It is popular on the Internet. Has a **.jpg** extension and scales nicely.
GIF	GIF (Graphic Interchange Format) is a compressed bitmap format originally developed by CompuServe. It supports up to 256 colors and is popular on the Internet. Has a **.gif** extension and scales nicely.

Image Boxes

An **image box** is very similar to a picture box in that it allows you to place graphics information on a form. Image boxes are more suited for static situations - that is, cases where no modifications will be done to the displayed graphics.

Image Box Properties:

Picture	Establishes the graphics file to display in the image box.
	If False, the image box resizes itself to fit the graphic.
	If True, the graphic resizes to fit the control area.

Image Box Events:

Click	Triggered when a image box is clicked.
DblClick	Triggered when a image box is double-clicked.

Drive List Box

The **drive list box** control allows a user to select a valid disk drive at run-time. It displays the available drives in a drop-down combo box. No code is needed to load a drive list box; Visual Basic does this for us. We use the box to get the current drive identification.

Drive List Box Properties:

Drive	Contains the name of the currently selected drive.

Drive List Box Events:

Change	Triggered whenever the user or program changes the drive selection.

Directory List Box

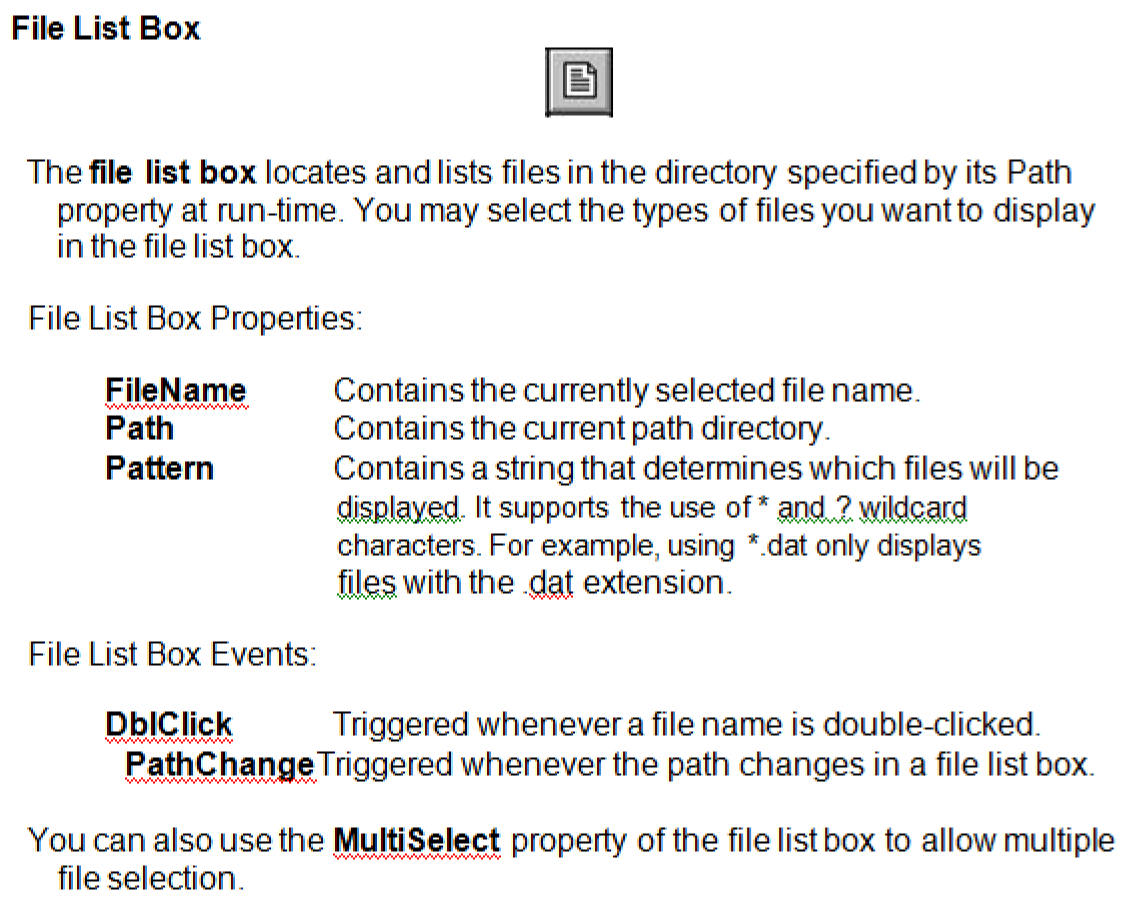
The **directory list box** displays an ordered, hierarchical list of the user's disk directories and subdirectories. The directory structure is displayed in a list box. Like, the drive list box, little coding is needed to use the directory list box - Visual Basic does most of the work for us.

Directory List Box Properties:

Path	Contains the current directory path.

Directory List Box Events:

Change	Triggered when the directory selection is changed.



File List Box

The **file list box** locates and lists files in the directory specified by its Path property at run-time. You may select the types of files you want to display in the file list box.

File List Box Properties:

FileName	Contains the currently selected file name.
Path	Contains the current path directory.
Pattern	Contains a string that determines which files will be displayed. It supports the use of * and ? wildcard characters. For example, using *.dat only displays files with the .dat extension.

File List Box Events:

DblClick	Triggered whenever a file name is double-clicked.
PathChange	Triggered whenever the path changes in a file list box.

You can also use the **MultiSelect** property of the file list box to allow multiple file selection.

7. What is MDI? What are the advantages of MDI forms and its features?

The Multiple Document Interface (MDI) was designed to simplify the exchange of information among documents; you can maintain multiple open windows.
The main Form, or MDI Form, isn't duplicated, but it acts as a container for all the windows, and it is called the parent window. The windows in which the individual documents are displayed are called Child windows. At run time, an MDI form and all of its child forms take on special characteristics:

- All child forms are displayed within the MDI form's workspace. The user can move and size child forms like any other form; however, they are restricted to this workspace.
- By setting the AutoShowChildren property, you can display child forms automatically when forms are loaded (True), or load child forms as hidden (False).

Advantages of MDI

- MDI applications can often handle multiple documents more readily than SDI programs.
  For example, many MDI text editors allow the user to open multiple text files side by side in the same window.
- It easy to compare and look up information from a second document while working on the first.
- MDI applications tend to perform more quickly than SDI programs

SDI	MDI
(i) It stands for single Document Interface.	(i) It stands for Multiple Document Interface.
(ii) There is no concept of parent window rather every window is independent of each other.	(ii) In MDI all the windows are a part of a larger parent window
(iii) It act as an container for other controls or objects	(iii) It acts as an container for other windows or forms.
(iv) No such type of property is available	(iv) It has MDI-child property which is used to setup a window as child.
(v) All the windows can be moved freely (i) It stands for single Document Interface. (ii) There is no concept of parent window	(v) The windows can be moved around only with the parent window. (i) It stands for Multiple Document Interface. (ii) In MDI all the windows are a part of a larger parent

8. Write short notes on the following – a) Activex Data Objects (ADO), b) Procedures and its types and c) Control Array

a) Activex Data Objects. (ADO)
Microsoft's latest set of data access objects are the ActiveX Data Objects (ADO). These objects let you access data in a database server through any OLE DB provider. ADO is intended to give you a consistent interface for working with a wide variety of data sources, from text files to ODBC relational databases to complex groups of databases.
The ADO object model consists of six objects:

- **Connection**: Represents an open connection, in this case to an OLE-DB data source that can be an ODBC data source, using MSDASQL (the Microsoft OLE-DB provider for ODBC).
- **Error**: Contains details about data access errors, refreshed for each time an error occurs in a single operation involving ADO.
- **Command**: Defines a specific command you wish to execute against data.
- **Parameters**: Optional collection off the command object for any parameters you wish to provide to the command.
- **Recordset**: Represents a set of records from a table, command object, or SQL Syntax. Can be created without any underlying Connection object.
- **Field**: Represents a single column of data in a recordset.
- **Property**: A collection of values raised by the provider for ADO.

b) Procedures and its types
A procedure is a set of one or more program statements that can be executed by referring to the procedure name. We can reuse the code written in a procedure, as it can be executed any number of times by making a call to the procedure.
The various types of Procedures are:
a) Sub procedure
b) Function procedure
c) Property procedure
In sub procedure has two types:

- General procedure
- Event handling procedure

General procedure	Event handling procedure	Function procedure	Property procedure
A general procedure is a block of code that performs a specific task. If we need to perform as specific times in different parts of the application, we can write the code in a procedure and call the procedure from anywhere within the application. It is a declared as public then other application can use it.	An Event handling procedure is a block of code that is executed when a specific event occurs, such as the click of button the loading of a form in the memory or fulfillment of a user defined condition. Example: If a user click a command button object with the name command1, the procedure named command1_click() is called and the code within the procedure is execute.	A function procedure is a block of code enclosed within the function. End function statements a function procedure unlike a sub procedure, returns a value to the calling code. If we need a procedure to returns a value so that the return value can be further procedure in the calling code, we should create a function procedure instead a sub procedure.	A property procedure is a set of code statements that are used to assign or creative the value of the properties declared with in a module, a class or a structure. Properties are type of variables that store the values for an object of a class or a structure.

c) Control Array:

Array of controls having same type & same name which share common properties is known as control array. They also share the same event procedures. Since controls have the same name VB needs a way to distinguish them. This is done by a property called index. This index property identifies the specific control & gives the flexibility to organize them at runtime. Control Array can be created either in design time or in runtime.

- Elements of the same control array have their own property settings.
- Common uses for control arrays included menu controls and option button groupings.
- If you want to create a new instance of a control at run time, the control must be a member of a control array

Using the control array mechanisms, each new control inherits the common event procedures already written for the array.

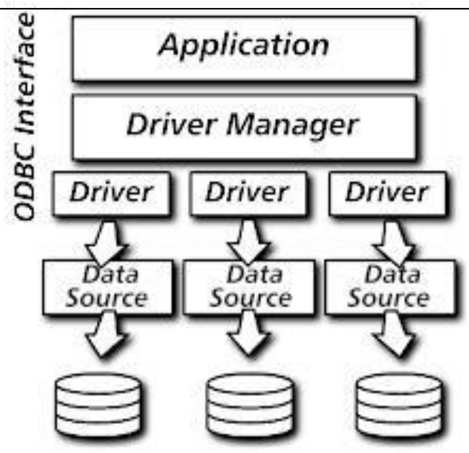
9. Briefly explain the ODBC architecture.

The ODBC (*Open Database Connectivity*) architecture has four components:

- **Application -** (Spreadsheet, Word processor, Data Access & Retrievable Tool, Development Language etc.) Performs processing by passing SQL Statements to and receiving results from the ODBC Driver Manager.
- **Driver Manager -** a *Dynamic Link Library* that Loads drivers on behalf of an application.
- **Driver** - a *Dynamic Link Library* that Processes ODBC function calls received from the Driver Manager, submitting the resultant SQL requests to a specific data source, and returns results to the application. If necessary, the driver modifies an application's request so

that the request conforms to syntax supported by the associated DBMS.

- **Data Source** Consists of a DBMS, the operating system the DBMS runs on, and the network (if any) used to access the DBMS.



The **Driver Manager** and **Driver** appear to an application as one unit that processes ODBC function calls.

<u>**Types of Drivers**</u>
ODBC defines two types of drivers:-
Single-tier - The driver processes both ODBC calls and SQL statements.
Multiple-tier - The driver processes ODBC calls and passes SQL statements to the data source**.**

10. What is module? Define class module, form module?
VB code that is not related to a specific form or control and which contains a procedure or variable that implements the common codes is known as modules (.bas).
Procedures that are to be used repeatedly in response to events in several different controls are placed in standard module hence it avoids duplication of code.

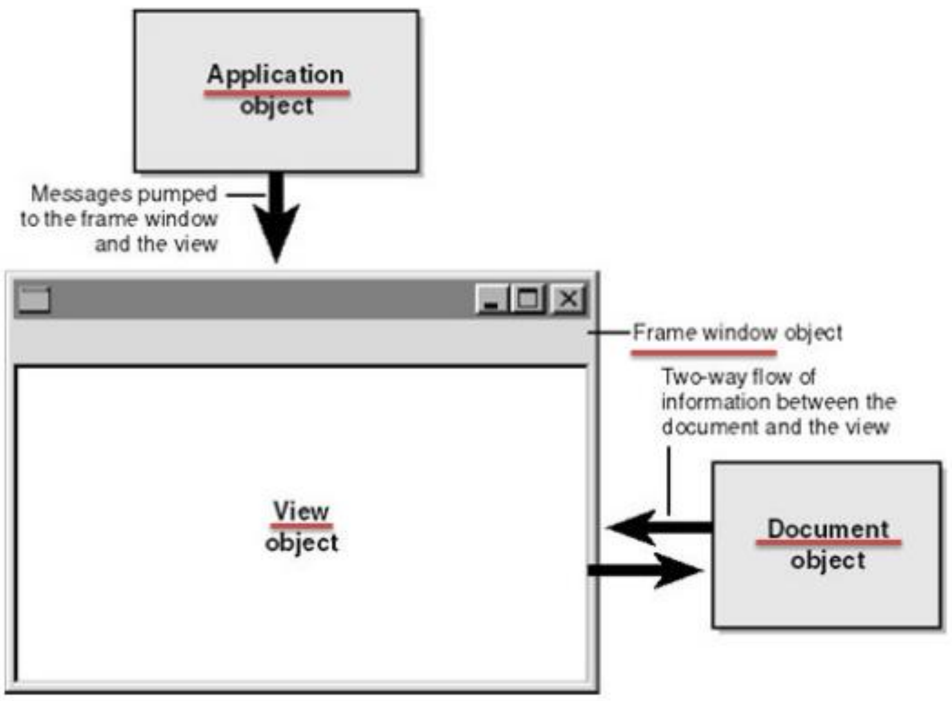
A module level variable can be accessible either within a module or used by all other modules. Variables used within a module are declared or private and the variable used by all other modules is declared as public.
There are three types of modules in VB:-
Form module, Standard modules, Class modules
The **class module** contains the defining characteristics of a class, including its properties and methods.
The **form module** contains the graphic elements of the VB application along with the instruction.
The **general module** generally contains general-purpose instructions not pertaining to anything graphic on-screen.

11. What is a Dynamic Link Library (DLL?)
All Windows applications at their most basic level interact with the computer environment by using calls to **dynamic link libraries** (**DLL**). DLL's are libraries of routines, usually written in C, C++, or Pascal, that you can link to and use at run -time.

Each DLL usually performs a specific function. By using DLL routines with Visual Basic, you are able to extend your application's capabilities by making use of the many hundreds of functions that make up the Windows Application Programming Interface (**Windows API**).
These functions are used by virtually every application to perform functions like

- ➢ displaying windows
- ➢ file manipulation
- ➢ printer control
- ➢ menus and dialog boxes
- ➢ multimedia
- ➢ string manipulation
- ➢ Graphics and managing memory.

:


Application object
Messages pumped to the frame window and the view
Frame window object
Two-way flow of information between the document and the view
View object
Document object

- Document/View Fundamentals Application object provides message loop and sends messages to the frame window and the view
- Frame Window object: –Shows menu, toolbar and status bar
- View object: –Displays (renders) data and translates mouse and keyboard input into commands to operate the data.
- Document object: –Stores data –Manipulates data

Document View Architecture

MFC Document View Architecture is a framework to manage user module data and view of the module separately. The design it follows is known as model-view architecture. Data part is managed by model and in MFC this is known as Document and display part is managed by view class.
The MFC document/view architecture includes a combination of a document in which data is stored and a view which has privileged access to the data.

Document

Document is the class to manage data part of the objects. Application often derives a class from CDocument. This deals with saving and retrieving data fields to and from the file stream.

View

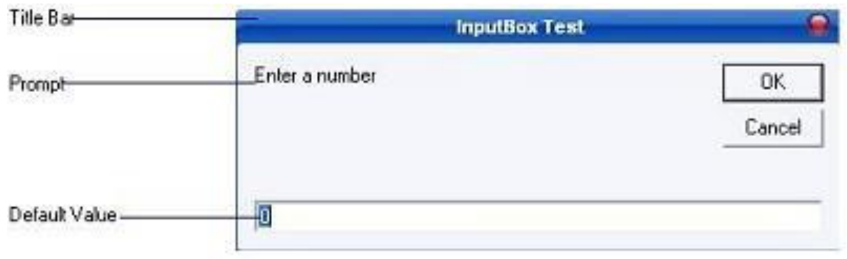
View is the class module to display data of the document. A derived class of CView is often used for this purpose. This display part can be the display of the document in graphical form or with UI elements or in the form of printable view which contains formatted text.

Frame

MFC framework uses a Frame window or class CFrame to display window in the screen and thus a CFrame class is always needed in any application which follows document view architecture.

13. Explain Input Box and Message Box functions.

Input Box :
Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns a String containing the contents of the text box.



Syntax :

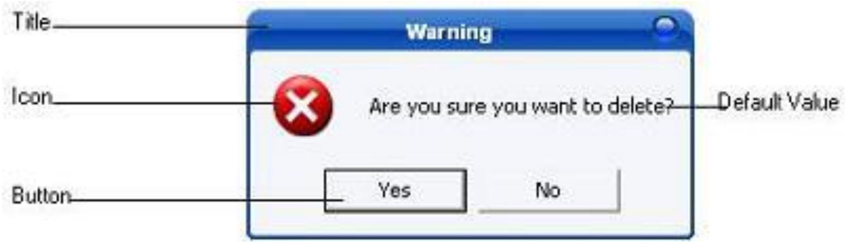
memory_variable = InputBox (prompt[,title][,default])

memory_variable is a variant data type but typically it is declared as string, which accept the message input by the users.

The arguments are explained as follows:

- **Prompt** - String expression displayed as the message in the dialog box. If prompt consists of more than one line, you can separate the lines using the vbCrLf constant
- **Title** - String expression displayed in the title bar of the dialog box. If you omit the title, the application name is displayed in the title bar
- **default-text** - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to key in.
- **x-position and y-position** - the position or the coordinate of the input box

Message Box:
Displays a message in a dialog box and wait for the user to click a button, and returns an integer indicating which button the user clicked.



Syntax :

MsgBox (Prompt [,icons+buttons] [,title] **)**

memory_variable = MsgBox (prompt [, icons+ buttons] [,title] **)**

Prompt : String expressions displayed as the message in the dialog box.
Icons + Buttons : Numeric expression that is the sum of values specifying the number and type of buttons and icon to display.
Title : String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.

Icons

Constant	Value	Description
vbCritical	16	Display Critical message icon
vbQuestion	32	Display Warning Query icon
vbExclamation	48	Display Warning message icon
vbInformation	64	Display information icon

Buttons

Constant	Value	Description
vbOkOnly	0	Display OK button only
vbOkCancel	1	Display OK and Cancel buttons
vbAbortRetryIgnore	2	Display Abort, Retry and Ignore buttons
vbYesNoCancel	3	Display Yes, No and Cancel buttons
vbYesNo	4	Display Yes and No buttons
vbRetryCancel	5	Display Retry and Cancel buttons

14. What do you mean by Connection String and Record Source in ADO data control?

Connection string property specify a data source by passing a detailed connection string containing a series of argument=value statement separated by semicolons.
ADO supports five arguments for the connection string property:

- Provider: = specify the name of a provider to use for the connection.
- File name: = specifies the name of a provider specific file (for example:= a persisted data source object) containing present connection information.
- Remote provider: = specifies the name of a provider to use when opening a client side connection. (Remote data service only)
- Remote server: = specifies the path name of the server to use when opening a client side connection. (Remote data service)
- URL: = Specifies connection string as an absolute URL identifying a resource , such as a file or a directory.

Connection string property is read/write when the connection is closed and read-only, when it is open. Record source:=
a) Gets or sets the statement or query that returns a Record set.
b) The Record source contains both the name of a database table and a valid SQL string using syntax appropriate for the data source.

15. What is Common Dialog Control? Give its methods

Adding a Windows Common Dialog control to our program is easy: the steps are as follows:

1. Select the Project|Components menu item.
2. Select the Control tab in the Components box that opens.
3. Select the entry labeled Microsoft Common Dialog Control, then click on OK to close the Components box.
4. Add a Common Dialog control to a form in the usual way_just double-click the Common Dialog tool in the toolbox, or select it and paint the control on the form.
5. Add the code we want to open the dialog box and make use of values the user sets.

<u>To display various dialog boxes, we use these Common Dialog methods:-</u>

CommonDialog1.ShowOpen_Show Open dialog box
CommonDialog1.ShowSave_Show Save As dialog box
CommonDialog1.ShowColor_Show Color dialog box
CommonDialog1.ShowFont_Show Font dialog box
CommonDialog1.ShowPrinter_Show Print or Print Options dialog box
CommonDialog1.ShowHelp_Invokes the Windows Help engine

<u>Common Dialog box:-</u> A Common Dialog Box is a window used to display and for accept information.

Dialog Boxes can either be modal or modeless.

- A Modal Dialog Box does not let the user continue working with rest of the application until it is closed.
- A Modeless Dialog Box, on the other hand lets the user shift the focus between the dialog box and another form without having to close the dialog box